Detecting Fabrication Faults in C-elements

O. A. Petlin, S. B. Furber
Department of Computer Science
University of Manchester
Manchester, M13 9PL, UK
email: {oleg, sfurber}@cs.man.ac.uk

Abstract - C-elements are used widely in asynchronous VLSI circuits. Fabrication faults in some C-elements can be undetectable by logic testing. Testable designs of static CMOS C-elements are given in this paper which provide for the detection of single line stuck-at and stuck-open faults. We show that driving the feedback transistors in the proposed testable static C-element transforms its sequential function into a combinational AND or OR function depending on the driving logic value. This simplifies the testing of asynchronous circuits which incorporate a large number of state holding elements. The scan testable C-element described can be used in scan testing of the asynchronous circuit making the states of its memory elements controllable and observable.

Key words - Asynchronous circuit, CMOS VLSI circuit, C-element, stuck-at fault, stuck-open fault, logic testing.

1. Introduction

Asynchronous or self-timed circuits have already demonstrated advantages over their synchronous counterparts. Some of the advantages are design flexibility, the absence of clock skew, the potential for lower power consumption and performance at the average speed rate rather than at the worst case [Hauck95], [Lav95]. A successful attempt to produce an asynchronous version of the ARM microprocessor has been reported [Furb94]. However, before producing a fully commercial asynchronous chip one must be sure that it is possible to show that it is fault-free after its fabrication. Testing asynchronous circuits is aggravated by the following factors [Hulg94]:

- the presence of a large number of state holding elements in asynchronous circuits makes the generation of tests harder or even impossible;
- detecting hazards and races is complicated;
- the absence of synchronization clocks decreases the level of test control over the circuit.

It has been observed that some classes of asynchronous circuits, such as delay-insensitive and speed independent circuits, are testable for a certain class of stuck-at faults. These circuits use handshaking protocols where each signal transition on a circuit line is acknowledged by another signal transition. In the presence of stuck-at faults such circuits halt. Asynchronous circuits are self-checking or self-diagnostic circuits if they exhibit no activity in the presence of stuck-at faults [Haz92], [Beere92], [David90].

The components used to design asynchronous circuits are complicated. Some of the fabrication faults inside asynchronous components are hard or even impossible to detect by logic testing. The main building block widely used in asynchronous VLSI circuits is the Muller C-element. *Brzozowski* and *Raahemifar* showed that the testing of line stuck-at faults in different implementations of the C-element is not trivial [Brzo95]. It has been observed that line stuck-at faults of the C-element fall into one of the following categories:

- faults that are detectable by logic testing since they halt the circuit or change its function;
- faults that are detectable by delay measurements;
- faults that may result in an oscillation;
- faults that may destroy the speed-independence of the circuit;
- faults that are detectable by measuring the circuit current.

In this paper we consider different CMOS implementations of static C-elements for testability. The C-element designs reported in this paper provide for the detection of line stuck-at and transistor stuck-open faults using logic testing. The structure of the paper is as follows: Section 2 discusses the testing of line stuck-at faults and transistor stuck-open faults in CMOS circuits; different implementations of static C-elements are examined in Section 3; Sections 4 and 5 present CMOS implementations of the C-element which are testable for stuck-open and stuck-at faults respectively; the C-element with scan features is considered in Section 6; cost implementation comparisons are made in Section 7; and, finally, Section 8 summarises the principal conclusions of the paper.

2. Testing for fabrication faults in CMOS circuits

Stuck-at and stuck-open fault models are used to describe the effects of the majority of fabrication faults in CMOS circuits [Weste93], [Wad78], [Red86], [Russ89]. The stuck-at fault model assumes that a fabrication failure causes the wire to be stuck permanently at a certain logical value. Consider a fragment of a CMOS design (in Figure 1a) with possible locations of line stuck-at faults. For instance, the stuck-at one fault in node 1 (1-SA1) is interpreted as a break on line 1 with the gate of *n*-type transistor N_2 connected permanently to the power supply voltage (in Figure 1b). The application of a constant voltage is marked with a cross. Fault 2-SA can be represented in three ways:

- the disconnection of transistor N_1 from node 2 and setting its source to a logical value (fault 2’-SA in Figure 1b);
- the disconnection of transistor N_3 from node 2 and setting its source to a logical value (fault 2’’-SA in Figure 1c);
- the disconnection of transistor N_2 from node 2 and setting its drain to a logical value (fault 2’’’-SA in Figure 1d).

Note that fault 2’’’-SA is equivalent to fault 1-SA0 when transistor N_2 is permanently off. Thus, fault 2’’’-SA can be excluded for the sake of simplicity. Notations 3’-SA or 3’’-SA denote a break on the left side of line *y* and setting a permanent logical value on its right

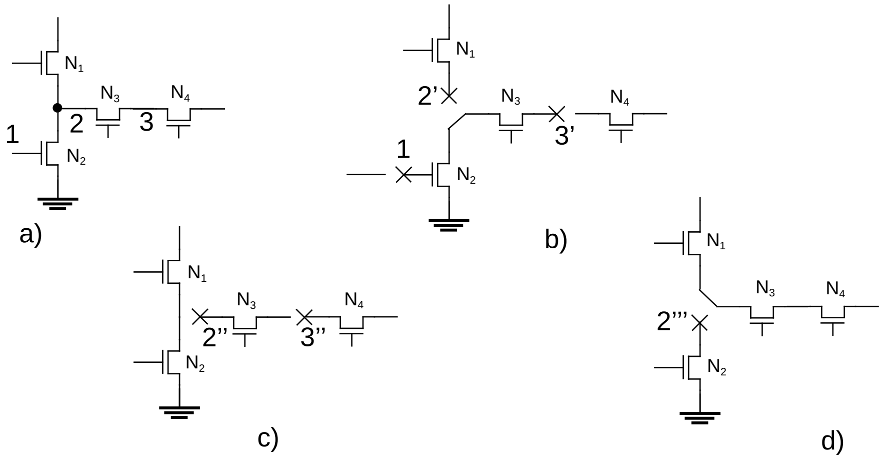

***Figure 1 :** Locations of line stuck-at faults and their interpretation in a fragment of CMOS design*

side or a break on the right side of line *y* and setting a permanent logical value on its left side respectively (compare Figures 1b and 1c).

The basic CMOS inverter shown in Figure 2a consists of two types of transistors: *p*-type and *n*-type transistors [Weste93]. When input *x* is low the *n* transistor is off and the *p* transistor is on. Output *y* is connected to the power supply voltage (V_{dd}) which corresponds to a logical one. If input *x* is high the *n* transistor is on and the *p* transistor is off. Output *y* is connected to ground (V_{ss}) which is a logical zero. Figure 2a shows line stuck-at fault locations in the CMOS inverter. For example, fault 1-SA1 of the inverter sets its output *y* to a constant logical zero. Input *x* of the inverter must be set to low to detect this fault, whereupon output *y* remains low whereas the fault-free response is high. Consider fault 2-SA0 in the inverter illustrated in Figure 2a. This fault sets transistor P_1 permanently on. If input *x* is high both transistors P_1 and N_1 are on. This leads to an uncertain situation when a logical one or zero can be registered by the test circuitry depending on the strengths of the transistors. As a consequence, the detection of fault 2-SA0 cannot be guaranteed by logic testing. Similar observations can be made for fault 3-SA1.

A stuck-open fault model represents a fault effect caused by a fabrication failure which permanently disconnects the transistor pin from the circuit node. Stuck-open faults can be opens on the gates, sources or drains of transistors. In the presence of a single stuck-open fault (SO) there is no path from the output of the circuit to either V_{dd} or V_{ss} through the faulty transistor. For example, in the presence of fault P_1-SO (Figure 2a) output *y* cannot beset high since there is no connection between V_{dd} and node *y*. This fault can be identified

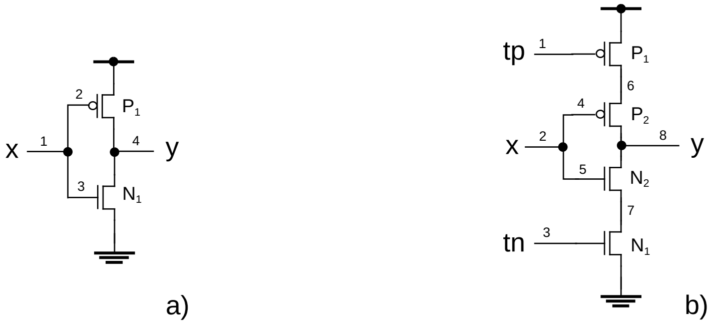

Figure 2 : *CMOS inverters: a) inverter with two logically untestable stuck-at faults; b) testable inverter*

by a set of two test patterns <T_1=1,T_2=0> applied sequentially to input *x*. As a result, the output of the faulty inverter remains low whereas the output of the fault-free inverter is high. Fault N_1-SO is detectable by a test set <T_1=0,T_2=1>.

Figure 2b shows a CMOS inverter which is testable for all single stuck-at and stuck-open faults. Two additional transistors (P_1 and N_1) controlled by two separate inputs are inserted into the inverter shown in Figure 2a. Table 1 contains tests for line stuck-at faults and transistor stuck-open faults in the inverter. Note that faults 4-SA0 and 5-SA1 are detectable by logic testing. For instance, a test sequence <T_1=111,T_2=100> applied to the inputs of the inverter detects fault 4-SA0. The effect of fault 6-SA0 or 7-SA1 is a ‘weak zero’ (0_w) or a ‘weak one’ (1_w) output signal respectively. These voltage levels are very close to the corresponding logical 1 and 0 voltage levels since output *y* was previously set to the same logical values. Faults 1-SA1 or 3-SA0 result in a ‘floating zero’ (0’) or ‘floating one’ (1’) output signal respectively. The output capacitance of the inverter can be considered as a dynamic memory element which keeps its precharged value for a certain time. It is assumed that the time between the application of two test vectors is small enough not to allow a floating output voltage level to reach the CMOS threshold level [Weste93], [Red86]. Hereafter we will treat weak and floating logical values as normal ones.

Table 1 : *Tests for stuck-at and stuck-open faults of the inverter in Figure 2b*

Single SA0 faults	Single SA1 faults	Single SO faults	Test sequences			Fault-free output	Faulty output
			x	tn	tp	y	y
2	8		1	1	0	0	1
8	2		0	1	0	1	0
	1,4	P_1,P_2	1	1	0	0	0
			0	1	0	1	0’
3,5		N_1,N_2	0	1	0	1	1
			1	1	0	0	1’
4			1	1	1	0	0
			1	0	0	0’	1
	5		0	0	0	1	1
			0	1	1	1’	0
1	6		1	1	1	0	0
			0	1	1	0’	1
7	3		0	0	0	1	1
			1	0	0	1’	0
6			1	1	0	0	0
			0	1	0	1	0_w
	7		0	1	0	1	1
			1	1	0	0	1_w

A stuck-at fault on the gate of a CMOS transistor keeps the transistor on or off permanently depending of the type of fault. Thus transistor stuck-open faults in CMOS designs can be represented by their correspondent gate stuck-at faults. For instance, fault 5-SA0 on the gate of transistor N_2 is equivalent to fault N_2-SO (see Figure 2b). As a result, testing for stuck-at faults of the inverter illustrated in Figure 2b guarantees the detection of all its stuck-open faults.

3. CMOS designs of the C-element

Figure 3a shows a symbolic representation of the two-input C-element with inputs *a*, *b* and output *c*. The C-element is a state holding element the output of which is high when its inputs are high and low when its inputs are low. Any other input combinations do not change the state of the C-element. The function performed by the two-input C-element can be written as follows:

$$c_t = a_t \cdot b_t + a_t \cdot c_{t-1} + b_t \cdot c_{t-1}, \qquad (1)$$

where c_t and c_{t-1} are the states of the C-element at time *t* and *t-1* respectively.

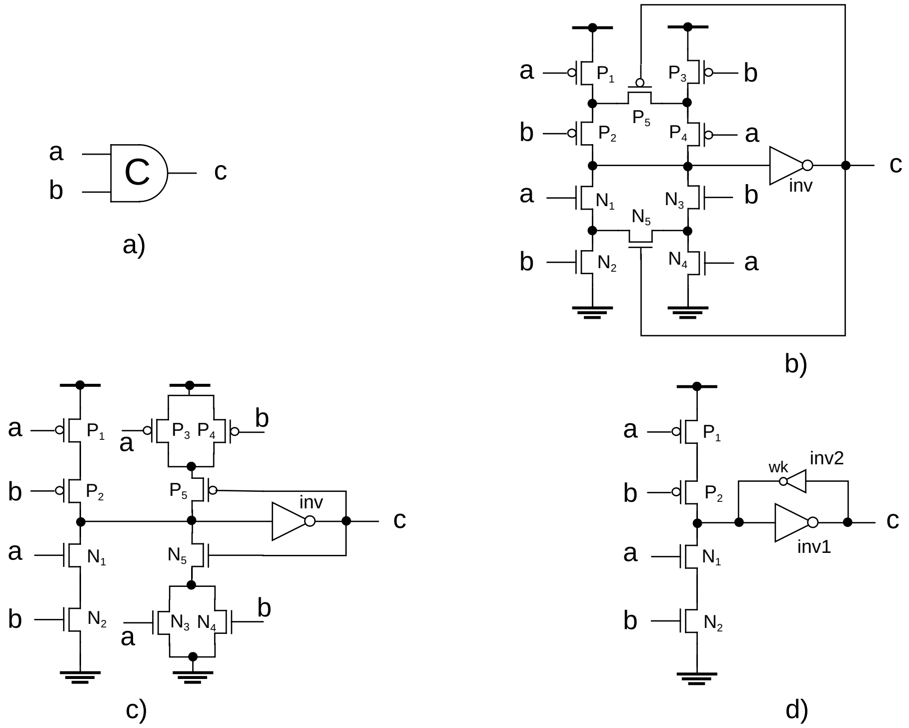

Figure 3 : *C-elements: a) symbol of the two-input C-element; b) and c) static C-elements; d) pseudo-static C-element.*

There are different ways to implement static C-elements in CMOS technology. Figure 3b shows a CMOS C-element which performs according to equation 1. For example, when *a*=1 and *b*=1 there is a path between V_{ss} and the input of inverter *inv*. As a result, output *c* of the C-element is high and feedback *n*-type transistor N_5 is on. If the inputs of the C-element are different there is always a connection between V_{ss} and the input of inverter *inv.*

Equation 1 can be rewritten in the following form:

$$c_t = (a_t + b_t) \cdot c_{t-1} + a_t \cdot b_t . \qquad (2)$$

A CMOS implementation of the C-element which performs according to equation 2 is illustrated in Figure 3c. This C-element works in a similar way as the one shown in Figure 3b. Both CMOS implementations of the C-element require 12 transistors.

The C-element shown in Figure 3d is a pseudo-static C-element which performs according to equation 1 but in a way similar to that of a dynamic C-element. The only difference is that the weak feedback inverter *inv2* is inserted into the C-element to create a CMOS memory. If c_{t-1}=0, a_t=1 and b_t=1 then the input of inverter *inv1* is driven to low since the strength of *n*-type input stack (transistors N_1 and N_2) is higher than that of *p*-type stack of weak inverter *inv2*. As a consequence, output *c* of the C-element goes high keeping the input of inverter *inv1* in low. If the input transistor stacks are disabled by different input signals the current state of the C-element is kept unchanged. The implementation of the pseudo-static C-element requires 8 transistors.

For test purposes we assume that the inputs of the C-element are controllable and its outputs are observable. It has already been shown in Section 2 that some of single stuck-at faults in the CMOS inverter are not detectable by logic testing. Therefore, such faults are not detectable in the CMOS designs depicted in Figure 3. There is a fundamental problem in the testing of static C-elements for stuck-open faults. Stuck-open faults in the feedback transistors of the static C-element transform it into a dynamic one. For instance, if the output of the weak inverter in the C-element shown in Figure 3d is disconnected from the input of inverter *inv1* the faulty C-element still performs according to equation 1, but as a dynamic circuit. This kind of fault can be identified only by ‘slow’ testing which ‘waits’ until the

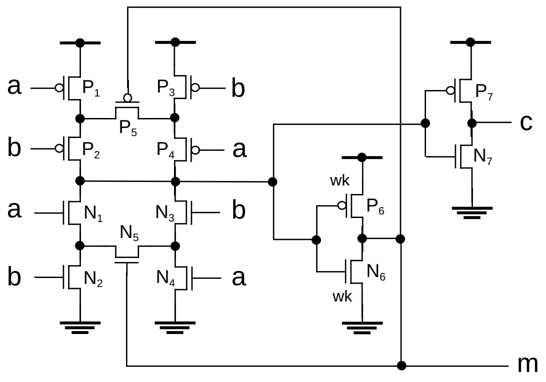

Figure 4 : *Locations of stuck-open faults in the static C-element*

output of the faulty C-element is discharged completely. This degrades the test performance. CMOS structures of the C-element which provide for the detection for single line stuck-at and transistor stuck-open faults are considered in the following sections.

4. Testing for stuck-open faults

The testing of stuck-open faults in the feedback transistors of the static CMOS C-element is implemented by driving them using an extra input and observing test results on its output. The structure shown in Figure 3b is most suitable as the starting point for the testable implementation. Figure 4 illustrates a CMOS design of the C-element where single transistor stuck-open faults are detectable. The C-element contains an additional weak inverter (transistors P_6 and N_6) the output of which can be overdriven by a logical value applied to pin *m*. The proposed implementation of the C-element requires 14 transistors.

Tests for the transistor stuck-open faults of the C-element are shown in Table 2. If m_i=0 or m_i=1 node *m* is used to drive feedback transistors P_5 and N_5 with a logical zero or one respectively. If m_i=z then node *m* is in a high impedance mode. A hyphen in the column headed m_o means that node *m* does not carry any diagnostic information since it is driven by an external logical value. It can be seen from Table 2 that faults P_5-SO and N_5-SO, which transform the static behaviour of the C-element into a dynamic one, are detectable by sets of two tests with no need for ‘slow’ testing.

Table 2 : *Tests for stuck-open faults of the C-element in Figure 4*

Stuck-open faults	Test sequences			Outputs of the fault-free C-element		Outputs of the faulty C-element	
	a	b	m_i	c	m_o	c	m_o
P_2,P_3,P_5,N_7	1	1	0	1	-	1	-
	1	0	0	0	-	1	-
P_1,P_4,P_5,N_7	1	1	0	1	-	1	-
	0	1	0	0	-	1	-
N_2,N_3,N_5,P_7	0	0	1	0	-	0	-
	0	1	1	1	-	0	-
N_1,N_4,N_5,P_7	0	0	1	0	-	0	-
	1	0	1	1	-	0	-
P_6	0	0	z	0	0	0	0
	1	1	z	1	1	1	0
N_6	1	1	z	1	1	1	1
	0	0	z	0	0	0	1

Driving the feedback transistors of the C-element transforms its sequential function into a combinational one depending on the logical value applied to pin *m*. Table 3 contains the operation modes of the C-element illustrated in Figure 4. When m_i=0 or m_i=1 the C-element is transformed into an AND or OR gate respectively. This transformation of the sequential function of the C-element allows a reduction in the number of state holding elements in the asynchronous circuit under test and makes its testing easier. On the other hand, once the circuit (in Figure 4) performs as the C-element output *m* can be used as a test point increasing its observability inside the asynchronous circuit.

Table 3 : *Operation modes of the C-element in Figure 4*

Function	Inputs			Outputs	
	a	b	m_i	c_t	m_o
AND	0	0	0	0	-
	0	1	0	0	-
	1	0	0	0	-
	1	1	0	1	-
OR	0	0	1	0	-
	0	1	1	1	-
	1	0	1	1	-
	1	1	1	1	-
Muller C	0	0	z	0	0
	0	1	z	c_{t-1}	c_{t-1}
	1	0	z	c_{t-1}	c_{t-1}
	1	1	z	1	1

The use of a weak inverter in the C-element illustrated in Figure 4 is not efficient in terms of power consumption in test mode. For instance, if during the test pin *m* is kept to one and a=b=0 then there is a path between V_{dd} and V_{ss} which increases the power dissipation of the C-element. The power consumption can be reduced if the weak inverter is replaced by a tristate inverter which is disabled during the test by an extra control input. It is easy to show that in this case such a C-element remains testable for transistor stuck-open faults.

5. Testing for stuck-at faults

The C-element illustrated in Figure 4 is not testable for line stuck-at faults since it has inverters which are not fully testable for all single line stuck-at faults (see Section 2). The implementation of the C-element shown in Figure 5 incorporates inverters which are testable for single stuck-at faults using two additional test inputs *tp* and *tn*. Two extra transistors controlled by inputs *tp* and *tn* are inserted in the feedback paths of the C-element for testability purposes. In normal operation mode, when *tp*=0 and *tn*=1 the circuit performs in the same manner as the one in Figure 4.

It is assumed that all the inputs and outputs of the C-element are controllable and observable during the test. Table 4 contains tests which are derived to detect its single line stuck-at faults. As was previously observed in Section 2 the detection of all single line stuck-at faults in a CMOS design guarantees the detection of all its single stuck-open faults. For instance, fault 18-SA1 is equivalent to keeping the appropriate *p*-type transistor off perma-

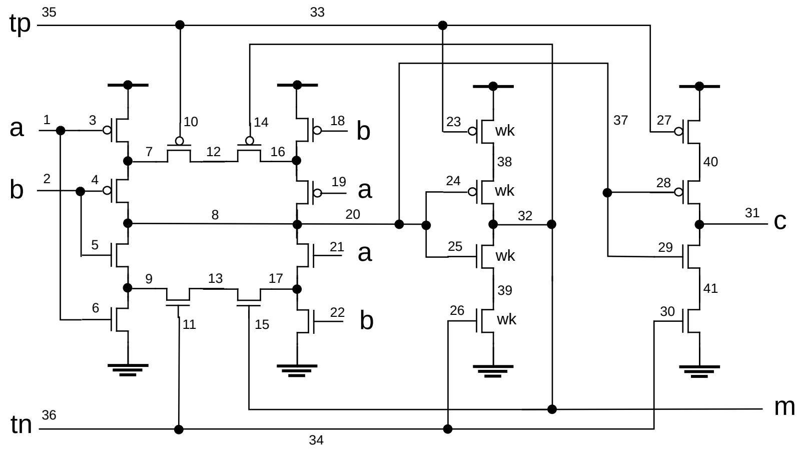

Figure 5 : *Locations of stuck-at faults in the static C-element*

nently which means its permanent disconnection from V_{dd}. As a result, the proposed CMOS design is testable for both stuck-at and stuck-open faults.

Table 4 : *Tests for stuck-at faults of the C-element in Figure 4*

No.	Single stuck-0 faults	Single stuck-1 faults	Test sequences			Outputs of the fault-free C-element		Outputs of the faulty C-element	
			a&b	tn&tp	m_i	c	m_o	c	m_o
1	31,32,38,40	8,20,37	11	10	z	1	1	0	0
2	8,20,37	31,32,39,41	00	10	z	0	0	1	1
3	7’,12’,16’,30	2,4,7”,10,12”, 14,18	11	10	0	1	-	1	-
			10	10	0	0	-	1	-
4	7”,12”,16”	1,3,10,14,12’, 16’,19	11	10	0	1	-	1	-
			01	10	0	0	-	1	-
5	2,5,9”,11,13”, 15,22	9’,13’,17’,27	00	10	1	0	-	0	-
			01	10	1	1	-	0	-
6	1,6,11,13’,15, 17’,21	9”,13”,17”	00	10	1	0	-	0	-
			10	10	1	1	-	0	-
7	25,26,29,30, 34,36		11	10	z	1	1	1	1
			00	10	z	0	0	1	1
8		23,24,27,28, 33,35	00	10	z	0	0	0	0
			11	10	z	1	1	0	0
9	4,10,18	16”	11	10	0	1	-	1	-
			01	11	0	1	-	0	-
10	3,10,19	7’	11	10	0	1	-	1	-
			10	11	0	1	-	0	-
11	17”	5,11,22	00	10	1	0	-	0	-
			10	00	1	0	-	1	-
12	9’	6,11,21	00	10	1	0	-	0	-
			01	00	1	0	-	1	-
13	14		11	00	1	1	-	1	-
			10	00	1	1	-	1	-
			10	11	0	1	-	0	-
14		15	00	11	0	0	-	0	-
			10	11	0	0	-	0	-
			10	00	1	0	-	1	-
15	23,27,33,35	38,40	00	10	z	0	0	0	0
			11	11	z	0	0	1	1
16	39,41	26,30,34,36	11	10	z	1	1	1	1
			00	00	z	1	1	0	0
17	24,28		00	11	z	0	0	0	0
			00	00	z	0	0	1	1
18		25,29	11	00	z	1	1	1	1
			11	11	z	1	1	0	0
19									

6. Scan testing of C elements

Scan testing has already become a standard methodology for testing VLSI circuits [Russ89]. Several reports describing different implementations of a scan test in asynchronous circuits have been published [Ron94], [Khoc94], [Pet95]. Scan testing presumes that the circuit is set to scan test mode where all its state holding elements are connected together forming a united scan chain. The scan path can be controlled either synchronously or asynchronously [Pet95]. As a consequence, the states of all memory elements are controllable and observable.

State holding elements of a scan testable circuit must operate at least in two modes: normal and scan test modes. In normal operation mode, the circuit performs according to its specification. During the scan test, the test patterns are loaded into the state holding elements and the test results are shifted out of the circuit. Figure 6 illustrates a CMOS implementation of the pseudo-static C-element with scan features. It contains two additional control inputs:

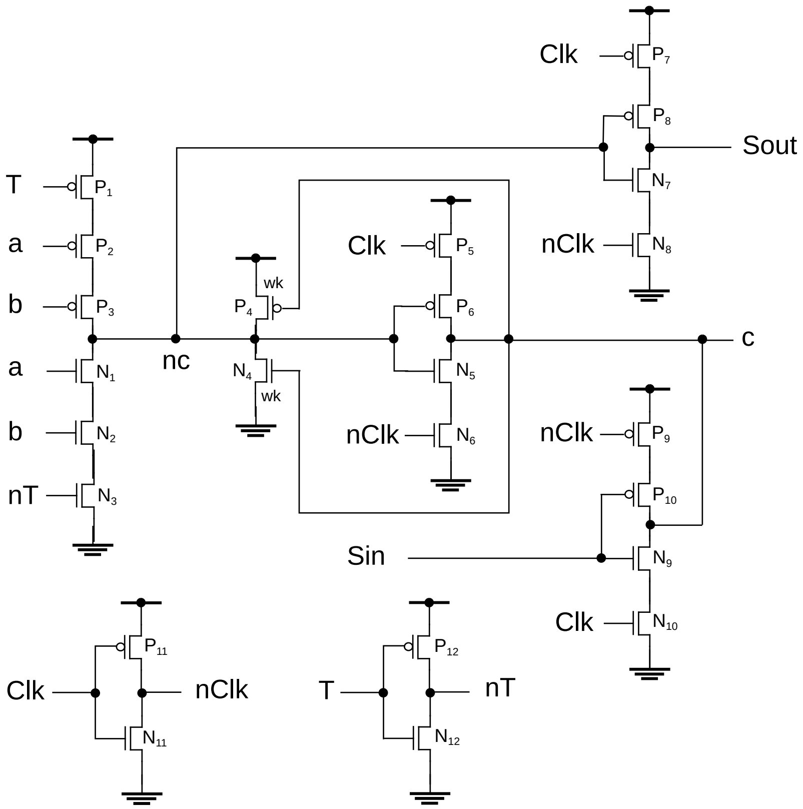

Figure 6 : *Pseudo-static C-element with scan features*

Table 5 : *Tests for stuck-open faults of the C-element in Figure 6*

Single stuck-open faults	Test sequences				Outputs of the fault-free C-element		Outputs of the faulty C-element	
	a&b	T	Clk	Sin	c	Sout	c	Sout
P_1-P_3,N_5,N_6	11	0	0	x	1	-	1	-
	00	0	0	x	0	-	1	-
P_5,P_6,P_{12},N_1-N_3	00	0	0	x	0	-	0	-
	11	0	0	x	1	-	0	-
P_7-P_{10},N_4,N_{11}	xx	1	1	1	-	x	-	x
	xx	1	0	1	-	0	-	0
	xx	1	1	0	-	1	-	0
	xx	1	0	0	-	1	-	0
P_4,P_{11},N_7-N_{10}	xx	1	1	0	-	x	-	x
	xx	1	0	0	-	1	-	1
	xx	1	1	1	-	0	-	1
	xx	1	0	1	-	0	-	1
N_{12}	00	0	0	x	0	0	0	0
	11	1	0	x	0	0	1	1

clock (*Clk*) and scan test (*T*) signals. Inputs *Sin* and *Sout* are used to scan the test pattern in and scan the state bit out of the C-element. Output *Sout* of each scan testable C-element (or any other scan testable memory block) is connected to input Sin of its successor forming the scan chain.

In normal operation mode, when *T*=0 and *Clk*=0 the C-element performs as the pseudo-static C-element depicted in Figure 3d. In scan mode, the input transistor stack is disabled by input *T* set to high. Clock signals are generated on input *Clk* to shift the test pattern from input *Sin* into the C-element. When signal *Clk* goes high the output transistor stack of the C-element is disabled and nodes *c* and *nc* are controlled from input *Sin*. Once the clock signal is low the negated bit loaded from input *Sin* is stored in the C-element and is passed to its output *Sout*. Clock signals generated on input *Clk* are used to shift the state bit of the C-element through the scan path to the test circuitry. When *Clk*=1 output *Sout* keeps its current logical value creating a dynamic memory and supplying input *Sin* of the following memory element. Clock signals must be kept high for enough time to guarantee the proper transmission of logical voltage levels.

An analysis of the C-element shown in Figure 6 reveals that it is testable for single transistor stuck-open faults. Table 5 contains tests for detecting stuck-open faults of the C-element. Symbol ‘x’ denotes a ‘don’t care’ signal. A hyphen means that the appropriate output is not used to observe the test results. Note that the fundamental problem of testing stuck-open faults in the weak feedback inverter of the pseudo-static C-element no longer exists since the weak transistors of the scan testable C-element participate in the scanning of the test data.

Consider an implementation of the scan testable CALL element in order to demonstrate how the C-element shown in Figure 6 can be used to build more complex scan testable asynchronous blocks. The CALL element is an event driven logic block [Suth89]. It remembers which of its inputs received an event first (*R1* or *R2*) and acknowledges the completion of the called procedure by an appropriate event on the matching output (*D1* or *D2*). The CALL element with scan features shown in Figure 6 performs using the two-phase signalling protocol where each signal transition denotes an event. All the inputs and outputs are initialized to zero. *T*=0 and *Clk*=0 in normal operation mode. When a rising request signal is generated on input *Ri* it primes C-element *Ci* and passes through the XOR gate producing a rising request signal on its output *R*. Once the required procedure completed an appropriate acknowledge event is generated on input *D*. As a result, C-element *Ci*

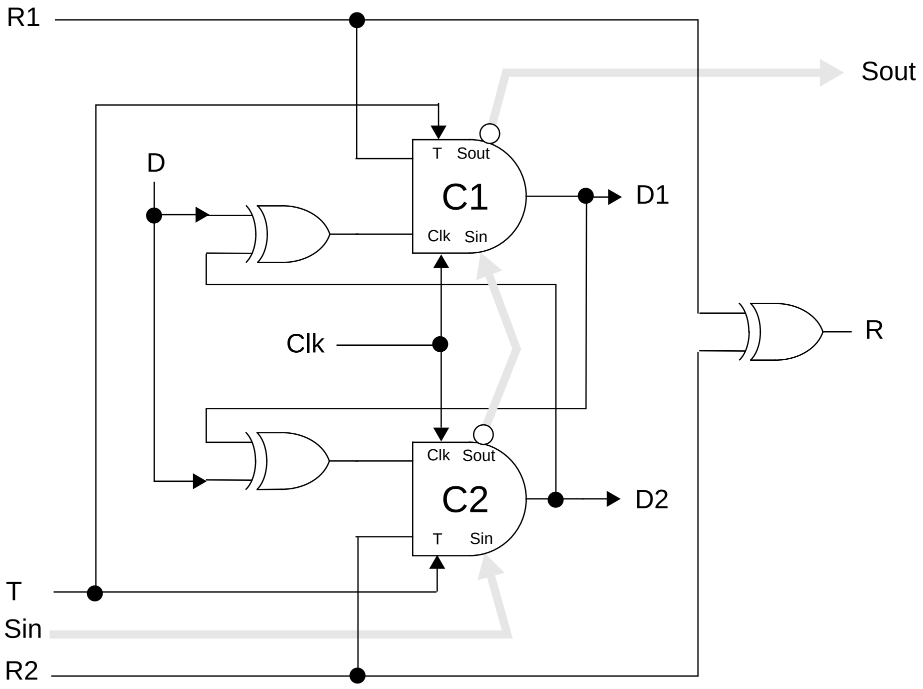

Figure 7 : *CALL element with scan features*

is set to one and a rising acknowledge signal is passed to output *Di* (i=1,2). The performance of the CALL element is identical for falling request events.

If the CALL element shown in Figure 7 is incorporated into an asynchronous VLSI circuit its internal states can be controllable and observable through the scan path. A test bit sent to input *Sin* of the C-element is negated on its outputs *Sout* and *c* (see Figure 6). The CALL element is tested by setting test control signal *T* to one. For instance, the clocked sequence 01 must be applied to input *Sin* of the CALL element in order to set its C-elements to one. The CALL element can perform its specified function when signals *T* and *Clk* are returned to zero. The state bits of the C-elements are shifted out of the CALL element and compared with known responses when its input *T* is set one and clocks are produced on its input *Clk*.

7. Cost comparisons

The testable structures of the C-element presented in the paper require different overheads depending on the fabrication faults to be detected. The designs of testable C-elements have been implemented on a $1\mu m$ CMOS process and their extracted layouts have been investigated using SPICE analyses [PeTR95]. Table 6 contains a summary of cost implementation comparisons of the CMOS C-elements and their testability. The largest number of transistors is required to implement the scan testable C-element. This is because scanning the data through the C-element can be implemented only in a master-slave manner which requires at least two memory elements. The implementation of the C-element shown in Figure 3 has just 17% overhead with one extra control input and guarantees the detection of all its stuck-open faults. The sequential function of such a C-element can be changed into a combinational one (AND or OR) which simplifies the testing of other components incorporated in the asynchronous circuit.

Table 6 : *Summary of costs of the testable C-elements*

Design	No. of transistors	No. of extra inputs/outputs	Overhead in No. of transistors	Layout overhead	Output nodal capacitance $\times 10^{-14} F$	Testability for faults
Figure 3	14	1	17%	17%	2.07	stuck-open
Figure 4	20	3	67%	45%	3.21	stuck-at & stuck-open
Figure 5	24	4	200%	115%	11.22	stuck-open

8. Conclusions

CMOS C-element designs for testability have been presented. Tests for detection of all single line stuck-at and transistor stuck-open faults of the proposed testable implementations of the C-element have been derived. It has been shown that although the CMOS C-element testable for line stuck-at faults allows the detection of all its single stuck-open faults its implementation requires more overhead than that of the C-element testable only for stuck-open faults. The fundamental problem of testing feedback transistors in static C-elements has been resolved. We illustrated that the sequential function of the C-element can be transformed into a combinational one (AND or OR) by driving its feedback transistors with an additional input. A scan test can be implemented by using the scan testable C-element design described in the paper. An analysis of the costs required to implement different testable C-element structures has been presented. It shows that the largest number of transistors is required to implement the scan testable C-element. The CMOS structures of the C-element considered in the paper can be used to design asynchronous VLSI circuits for testability.

Acknowledgements

We would like to express our gratitude to Jim Garside and John Brzozowski for their useful comments and discussions on the topic of this paper. Special thanks to Craig Farnswoth for his technical assistance with *Cadence* CAD tools.

References

[Beere92] P. A. Beerel, T. H. Y. Meng, “Semi-modularity and testability of speed-independent circuits,” Integration, The VLSI Journal, 13, 1992, pp. 301-322.

[Brzo95] J. A. Brzozowski, K. Raahemifar, “Testing C-elements is not elementary,” Proc. 2nd Working Conf. on Asynchronous Design Methodologies, South Bank University, May 30-31 1995, pp. 150-159.

[David90] I. David, R. Ginosar, M. Yoeli, “Self-timed is self-diagnostic,” TR-UT-84112, Department of Computer Science, University of Utah, Salt Lake City, UT, USA, 1990.

[Furb94] S. B. Furber, P. Day, J. D. Garside, N. C. Paver, J. V. Woods, “AMULET1: A micropipelined ARM,” Proc. IEEE Comput. Conf., March 1994.

[Hauck95] S. Hauck, “Asynchronous design methodologies: An overview,” Proc. IEEE, Vol. 83, No. 1, Jan. 1995, pp. 69-93.

[Haz92] P. Hazewindus, “Testing delay-insensitive circuits,” Ph.D. thesis, Caltech-CS-TR-92-14, California Institute of Technology, 1992.

[Hulg94] H. Hulgaard, S. M. Burns, G. Borriello, “Testing asynchronous circuits: A survey,” TR-FR-35, Department of Computer Science, University of Washington, Seattle, WA, USA, 1994.

[Khoc94] A. Khoche, E. Brunvand, “Testing micropipelines,” Proc. Int. Symposium on Advanced Research in Asynchronous Circuits and Systems (Async94), Utah, Nov. 1994, pp. 239-246.

[Lav95] L. Lavagno, A. Sangiovanni-Vincentelli, “Algorithms for synthesis and testing of asynchronous circuits,” Kluwer Academic Publishers, 1993.

[Pet95] O.A.Petlin, S.B.Furber, “Scan testing of micropipelines”, Proc. 13th IEEE VLSI Test Symposium, Princeton, New Jersey, USA, May 1995, pp. 296-301.

[PeTR95] O.A.Petlin, S.B.Furber, “Designing C-elements for testability”, Technical Report UMCS-99-10-2, Department of Computer Science, University of Mancheter, Manchester, UK, 1995.

[Red86] M. K. Reddy, S. M. Reddy, “Detecting FET stuck-open faults in CMOS latches and flip-flops,” IEEE Design & Test of Computers, vol. 3, no. 5, Oct. 1986, pp. 17-26.

[Ron94] M. Roncken, “Partial scan test for asynchronous circuits illustrated on a DCC error corrector,” in Proc. Int. Symposium on Advanced Research in Asynchronous Circuits and Systems (Async94), Nov. 1994, pp. 247-256.

[Russ89] G. Russell, I. L. Sayers, “Advanced simulation and test methodologies for VLSI design,” Van Nostrand Reinhold (International), 1989.

[Suth89] I. E. Sutherland, “Micropipelines,” Communications of the ACM, Vol.32, no.6, June 1989, pp. 720-738.

[Wad78] R. L. Wadsack, “Fault modelling and logic simulation of CMOS and MOS circuits,” Bell System Tech. Journal, vol. 57, May-June 1978, pp. 1449-1474.

[Weste93] N. H. E. Weste, K. Eshraghian, “Principles of CMOS VLSI design: A systems perspective,” Addison-Wesley Publishing Co., 1993.